HTML5

Tvoříme dynamické aplikace

BY

Jan Barášek (Baraja)



Drazí čtenáři,
Právě držíte v ruce můj několika měsíční projekt. Jedná se o knižní manuál (příručku), která má svým obsahem pomáhat jak začátečníkům, tak i pokročilým. V tomto vydání se budu zaobírat jazyky HTML5, PHP a propojení těchto dvou úžasných technologií.

Hned na úvod bych chtěl poznamenat, že kniha obsahuje opravdu mnoho ukázek kódů a hotových řešení. Pokud se vám nějaká část programu zalíbí, tak si jí můžete rozhodně zkopírovat a vložit právě do vaší webové aplikace.

Licence vydání

Toto dílo je v úplném základu šířeno zdarma (elektronicky), ale existují i tištěné verze. Výtisk si můžete koupit za opravdu minimální cenu (jen náklady na tisk a doručení) a získáte tak offline pomocníka do dob, kdy nemůžete být na počítači a nebo na nějaké jiné čtečce.

Zároveň každý platící čtenář dostane jako dárek webový prostor pro testování běhu vašich začátečnických aplikací. Jedná se o subdoménu na doméně baraja.cz s prostorem 500MB, bez reklam a s nainstalovaným PHP modulem.
Upozorňuji, že se jedná o prostor výhradně pro testování aplikací a nebo umístění jednoduchých stránek, rozhodně zde nespouštějte náročné procesy a cyklické scripty. V takovém případě vám bude prostor pro web zrušen.

Přispějte také!

Rádi píšete články a myslíte si, že máte co říct? Připojte se do tvorby dalšího vydání této originální knihy, o které se nebojím prohlásit, že se jedná o první českou otevřenou knihu o programování v tištěném tak elektronickém vydání.

Stáhněte si vše potřebné

Přikládání CD médií a nebo jiných datových nosičů se při distribuci této publikace příliš nedaří, proto byl vytvořen speciální webový server, odkud si můžete vše zdarma stáhnout a tím ušetřit jak práci nakladatelům, tak i sami sobě.

Vše je volně stažitelné na adrese: http://html5.baraja.cz

Současná nabídka programů se neustále rozšiřuje. V době vydání této knihy je HTML5 stále v bouřlivém vývoji a nikdo pořádně neví, jak vše dopadne.
Na webu si stáhněte virtuální server pro testování PHP skriptů přímo ve vašem počítači. Pokud tu možnost nemáte, využijte online úložiště (viz výše).

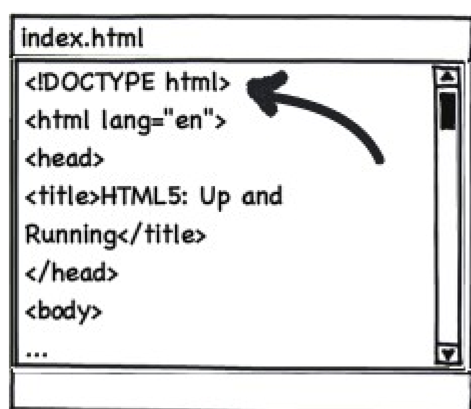
Použitý software autorů

Všechen zdejší obsah byl vytvořen za pomoci otevřených technologií, které jsou zdarma k dispozici. Kniha byla napsána v programu LibreOffice Writer v operačním systému Linux, distribuce Ubuntu ve verzi 10.04, 11.10 a 12.04.
Všechny programy jsou v základní instalaci Ubuntu.
Ubuntu je volně stažitelné na stránkách http://ubuntu.cz

OpenSource zdar!

Jak to vlastně začalo

Všechno to začalo celkem dávno. V roce 1989 člověk jménem Tim Berners-Lee dostal geniální nápad. V té době totiž internet jako takový dávno existoval, ale všechna komunikace probíhala formou příkazového řádku, který je podobný dnešnímu terminálu u Linuxu a posílali se příkazy. Tima napadlo udělat něco lepšího, co se bude snadno ovládat a bude to hlavně univerzální pro všechny platformy a různé druhy zařízení.
Vznikl jazyk HTML.
První verze toho moc neuměla. Vlastně to byly obyčejné bílé plochy na kterých byl černý text a sem-tam nějaký odkaz, toť vše. Sice to neumělo žádný grafický režim a podobalo se to stále terminálu, tak to ale bylo přeci-jen jiné. Text nebyl jen prostě vygenerovaný a poslaný jako odpověď na příkaz, ale byla to samostatná stránka, která měla svojí pevnou adresu v síťi (URL) a byla obohacena o značky.
V minulosti se všechny HTML značky psaly velkými písmeny a byly uzavřeny do závorek jako dnes. Že je to hodně podobné? Ano, základní kořeny jsou přes 25 let staré. Jsou jiné, a přeci stále stejné.



Celý jazyk je tedy složen vlastně ze značek. Kód stránky není nic jiného než prostý text, do kterého se vkládají šikmé závorky. Nejprimitivnější příklad:

```
<b>Já jsem tučný text</b>
<i>Já  jsem šikmý text</i>
<u>A já jsem podtržený text, nikoliv odkaz</u>
```

Je dobré poznamenat, že některé značky končí lomítkem a zopakováním toho samého „slova“.

Pár věcí, co musíte vědět

U začátečníků panuje při prvním seznámením s jazykem plno stejných dotazů. Zde jsou nejzákladnější informace o všem, co je potřeba k začátku vývoje.

- Do úplných začátků není potřeba vůbec internetové připojení
- Vystačíte si s běžným počítačem, není potřeba žádný velký výkon
- K psaní kódu stačí jakýkoliv textový editor (př. Poznámkový blok, bluefish, Pspad, Notepad++)
- Kódy rozhodně nepište ve Wordu a programech tomu podobným
- Hodí se mít kamaráda, co tomu rozumí a případně poradí. Když nikoho takového nemáte, tak využijte služeb nějakého webového fóra (na př. http://diskuse.jakpsatweb.cz)
- Hodí se mít víc internetových prohlížečů, protože se v každém zobrazuje stránka lehce jinak
- Kódy, skripty, tagy nejsou jako vzorečky v matematice. Neučte se je z paměti. Správný programátor si nemá pamatovat tisíce řádků kódu, ale má je umět vymyslet
- Když uděláte chybu a nevíte co s tím, využijte validátor kódu (český nebo oficiální anglický)
- Samotný jazyk je naprosto zdarma a za jeho použití se nikde neplatí
- PHP není jen vylepšené HTML!!! Je to úplně něco jiného, ale o tom později
- Obyčejné HTML stránky na disku jde otevírat i bez připojení na internet
- Snažte se vyhnout editorům typu „Microsoft Office FrontPage“ a podobným

Začínáme

Upozornění: V základním kurzu se jedná o vysvětlení základních funkčností. Víceméně to je HTML4 a samé obecné zápisy. K HTML5 se dostanu až později.

Úplně obyčejnou stránku uděláte tak, že jednoduše napíšete text do editoru, uložíte jako soubor *.html, otevřete v prohlížeči a text se zobrazí. Ale to není to pravé ořechové. Toto je vlastně jen úplně obyčejný způsob, jak obyčejně vypsat text a dost se to podobá stránkám v době vzniku internetu.
Pro začátek nečekejte, že z vás tato kniha udělá za víkend programátora, nicméně vás naučí ty největší základy, bez kterých se jen těžko obejdete.

Celý dokument (stránka) se skládá ze značek. To jsou takové ty kódy uzavřené do šikmých závorek. Jednoduše by se dalo říci, že se tyto značky píší za sebe a pak to vytvoří stránku. Není to ale jen o kopírování.

Každá stránka má nějakou hlavičku, obsah a patičku. Ten kdo se v kódech moc nehrabe asi neví, která bije. Jde o to, že každá stránka musí mít (správně by měla) obsahovat nějaký titulek, uvedení kódování znaků (hodně důležité) a podobně.
Nezapomeňte se všimnout zvýrazněných míst v kódu.

Příklad:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
 <html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8“>
  <title>Titulek stránky</title>
 </head>
 <body>
Sem můžete napsat jakékoliv značky a bude to fungovat.
 </body>
 </html>
```

Je to na vás příliš složité a dlouhé? Zvykejte si! Zkrátit se to nedá a vaše kódy budou mít jednou i stovky řádků.
Pozornější čtenář si jistě všiml, že kód obsahuje jak párové, tak nepárové značky. Párová značka je taková, která je v kódu 2x. Jednou normálně, podruhé s lomítkem. Ukázkou je třeba značka <body>.

Celý dokument je rozdělený na několik hlavních částí, a to převážně: Úvod (Doctype), Hlavička (Kódování, titulek) a tělo dokumentu (to co se zobrazí).
Doctype ujednává o tom, že se jedná o HTML dokument standartu 4.01, o to se příliš starat nemusíte.
V hlavičce jsou tagy celkem 2, ale může jich být až pár desítek. První pojednává o tom, že je jedná o soubor ve formátu .html napsaný formou textu v kódování UTF-8 (nejrozšířenější, doporučuji).
Druhý je nadpis stránky. To se např. Zobrazí ve výsledcích hledání, když budete hledat vaší stránku ve vyhledávači. Doporučuji zadat optimálně 2-5 slov, které plně vystihují váš web.
V obsahu stránky zatím je jen ilustrační text, tam se píše vše ostatní.
Až budeme tedy tvořit jakoukoliv stránku, tak zachováme takovouto šablonu a pak budeme upravovat jen obsah mezi <body> a </body>.

Jaké všechny značky existují?

HTML zná už několik desítek, možná stovek značek. Na celé této straně najdete v pravé části jejich abecední seznam.
Průměrný vývojář si jich pamatuje přibližně 50, víc to ani nemá význam. Ono to totiž není pokaždé jen napsat daný tag do závorek, ale většina z nich obsahuje také další atributy (nastavení), které daný objekt více popisují a tím zpřesňují jeho estetický vzhled. Pamatovat si to celé z paměti asi ani nemá význam, když tu máte tuto příručku. Já sám si pamatuji z paměti jen kolem 40 nejpoužívanějších.

Atributy

Většina značek se dá dále rozšiřovat a upřesňovat, ale to je každému snad jasné. Nejlepší příklad bude třeba na obrázku.



Toto je originál obrázku. Má velikost 128x128px, tento kód ho vykreslí v originále, stejně jak ho vidíte zde.

<img src="planeta.png">

Tag IMG má jeden atribut. Tím je SRC, které definuje adresu (cestu o obrázku). SRC jako Source (zdroj).



Tentokrát byl zmenšen atribut obrázku, který vyjadřuje výšku. Pokud by se změnil jen jeden z atributů rozměrů, tak by se druhý dopočítal a nešlo by k deformaci. Tentokrát jsem ale změnil oba, takže se planetka zploštila.

<img src="planeta.png" width="128" height="64">

Co jaký parametr znamená je jasné z ilustračního obrázku.

Toto však nejsou všechny atributy. Každý objekt (v našem případě obrázek), může mít desítky různých atributů, a některé se mohou dokonce opakovat.

TIP: Dobré je k obrázkům přidat atribut TITLE a ALT
<img src="planeta.png" alt="nějaký popisek pro vyhledávače. Podle něj se pak bude zobrazovat v hledání obrázků" title="toto se zobrazí v bublině, když ukážu na obrázek myší">

<!-- -->
<!--[if ..]-->
!doctype
&entity;
a
abbr
acronym
address
applet
area
b
base
basefont
bgsound
big
blink
blockquote
body
br
button
caption
center
cite
code
col
colgroup
dd
del
dfn
dir
div
dl
dt
em
embed
fieldset
font
form
frame
frameset
h1
h2
h3
h4
h5
h6
head
hr
html
i
iframe
img
input
ins
kbd
label
layer
legend
li
link
map
marquee
menu
meta
multicol
nobr
noembed
noframes
noscript
object
ol
optgroup
option
p
param
pre
s
samp
script
select
small
spacer
span
strike
strong
style
sub
sup
table
tbody
td
textarea
tfoot
th
thead
title
tr
tt
u
ul
var
wbr

Odkazy

Aneb nejdůležitější část stránky...

Znáte stránky bez odkazů? Já tedy ne. Odkazy jsou základem každé správné stránky. Odkaz se dá navádět jen jako prostý text a nebo ho může nahradit klikací objekt, např. Obrázek.

Odkaz na ubuntu
Strana 2

Jaký je mezi tím rozdíl? První odkaz vede na externí stránku na internetu a druhý v rámci domény. Tedy po vašich stránkách a nebo pevném disku.
Odkazování formou obrázku také není vůbec složité. Jak jistě víte, tak můžeme značky kombinovat téměř jak se nám zlíbí. Proto použiji tento kód odkazu a obrázku z předchozí strany.

<img src="planeta.png" title="Mapy Google!">

V prohlížeči se to zobrazí jako běžný obrázek, ale když na něj kliknete, tak se ocitnete na stránkách map od Googlu.

Musí tam být to http://?

U odkazu na externí webovou stránku do internetu ano! Jinak by to nefungovalo.

Nadpisy a formátování

Každý mi dá za pravdu, že je důležité dobře umět nastavit písmo, barvu a tak. V HTML to je velice snadné. Prohlédněte si stránku příkladů:

Toto je červený text
Toto je tučný text, *<i>toto zase šikmý</i>* a <u>toto zase podtržený</u>

<h1>Toto je největší možný nadpis</h1>

<h2>Toto je poněkud menší nadpis</h2>

<h3>Toto je střední nadpisek...</h3>

<h6>Toto je nejmenší nadpis, ještě existuje H4 a H5</h6>

<center>Tento text bude vystředěný uprostřed monitoru</center>

Dost ukázek, zbytek si každý vyzkouší sám podle přehledu na straně 5.

Je také dobré poznamenat, že mezera (nový řádek) se nedělá tak, že v kódu zmáčknete ENTER!
Je na to speciální značka a jmenuje se **
**

H T M L 5

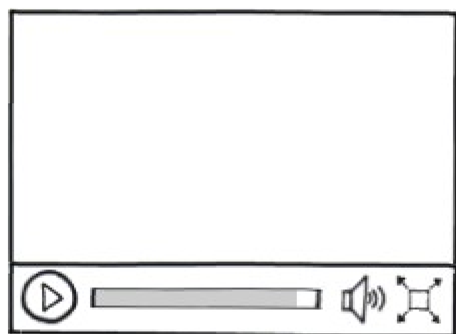
Základů už bylo dost. Pokud s HTML ještě moc neumíte, tak následující texty pro vás nejsou příliš vhodné. Tato kniha je převážně o HTML5, PHP a propojení mezi sebou.
Prvních několik stran úvodu mělo seznámit nováčky s tím, jak to vlastně funguje. Pokud se tedy chcete pustit rovnou do HTML5, tak můžete, ale opatrně.

Upgrade

„Upgrade“ na HTML5 může být tak snadný, jak snadno dokážete změnit doctype vaší stránky. Doctype byste již měli mít na prvním řádku každé stránky v HTML. Předchozí verze HTML definovaly řadu různých variant doctype a vybrat z nich ten správný nebylo vždy tak úplně snadné. V HTML5 najdete jen jeden doctype:

<!DOCTYPE html>

„Změna jazyka“ nerozbije váš kód. Všechny značky z HTML4 vám budou fungovat nadále. HTML5 vlastně jen přidává spoustu nových značek a možností usnadnění.

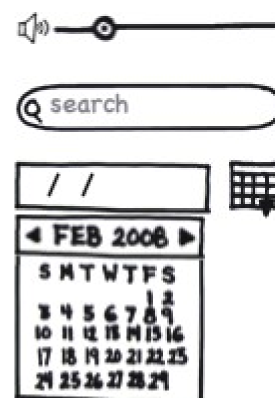


HTML5 není jen pár nových značek, získáte úžasné možnosti. Na př. Do dnes jste museli pokaždé video vkládat přes nějaký flashový přehrávač a nebo uploadovat na server typu YouTube. Nyní stačí jen použít pár nových tagů a můžete přehrávat přímo zdrojový soubor videa ve vašem prohlížeči.

Ono to je všechno krásné, dynamické, hezky funkční, ale podpora je zatím celkem slabá.

Nemusíte zahazovat kód, který už máte napsaný. Nemusíte se znovu učit, co jste se už jednou naučili. Pokud vaše webová aplikace včera fungovala jako HTML 4, bude dnes stále fungovat jako HTML5.

Oni se vlastně používají pořád ty samé kódy a jen se přibaluje balíček nových elementů. Podívejte se na ilustrační obrázek a uvidíte, co vše HTML4 prostě neumí a musí se to řešit složitě přes Javascript, Flash a podobně.



Říká se, že je podpora v dnešní době slabá. Není. Na př. Canvas dnes funguje dobře ve všech důležitějších prohlížečích. Nejlepší podporu má HTML5 zatím převážně v Opeře, Chromu a Firefoxu. Pokud chcete začít, doporučuji začínat na Opeře, která má podporu ze všech prohlížečů nejlepší.

Detekce podpory

Než se do toho pustíte, tak je dobré zkontrolovat, zda váš prohlížeč vůbec HTML5 zvládá. Neexistuje žádný obecný kód, který by napsal zda je prohlížeč kontabilní nebo ne, ale dá se to udělat jinak. Každý HTML objekt se dá spustit a zjistit, zda se vykreslil. Toto řeší poměrně dobře javascriptová knihovna Modernizr. Vždy nejnovější verzi si můžete stáhnout na stránkách http://modernizr.com/
TIP: Modernizr umí ověřit i podporu některých funkcí CSS3, což se může leckomu hodit.

Pokud chcete tedy knihovnu použít, stačí vložit do oblasti <head> vaší stránky následující zápis:

```
<head>
  <meta charset="utf-8">
  <title>Kontrola funkčnosti HTML5 přes knihovnu Modernizr</title>
  <script src="modernizr.min.js"></script>
</head>
```

Všechno se to spouští automaticky, takže se nemusíte o nic starat. Po spuštění vás může zajímat vždy nějaký konkrétní element. Hodí se vždy udělat podmínku. Pokud bude element podporován, tak se vykreslí, když ne, tak holt ne a dá se to vyřešit jinak.

```
if (Modernizr.canvas) {
  // sem dejte nějaký kód, protože je Canvas podporovaný!
} else {
  // Toto se vykreslí, když zrovna Canvas nemáte. Můžete sem dát např. Obrázek formou souboru.
}
```

Značka <canvas> se v HTML5 definuje vlastně jako prázdný obdélník ve kterém nic není (kreslící plátno) a do kterého můžete za pomoci spousty funkcí vykreslovat čáry, křivky, kolečka, obdélníky, čtverečky a vše co si umanete. Jedná se vlastně jakoby o vektorovou grafiku.
Canvas běží v reálném čase a také se tak může měnit. Proto můžete vykreslit téměř cokoliv a pak to postupně měnit. Proto můžete celkem jednoduše vytvořit jednoduchou aplikaci na úrovni ručičkových hodinek až po složitou hru, kde by si každý myslel, že se jedná o flash.

Canvasový úvod

Jako první ukázku uvedu ten nejzákladnější skript pro canvas. Nic to nevykreslí, běží to jen na pozadí prohlížeče. Nikdo ho nikdy neuvidí, tedy alespoň ne do doby, než to bude potřeba.

```
function supports_canvas() {
  return !!document.createElement('canvas').getContext;
}
```

Podpora prohlížečů:

IE	Firefox	Safari	Chrome	Opera	iPhone	Android
7,0+	3.0+	3.0+	3.0+	10.0+	1.0+	1.0+

Internet Explorer 7 a 8 vyžadují explorercanvas knihovny třetích stran. Internet Explorer 9 podporuje <canvas> nativně.

Živý canvas

Jak vlastně canvas vypadá v základním stavu? Nijak, je to prostě jen bílá plocha, dokonce nemá ani žádný rámeček. Dalo by se to přirovnat např. Tagu <div> nebo <span>

Vidíte canvas?



Že ne? Ale ano, vážně tam je. Akorát je průhledný.

Když to hodně zjednoduším, tak úplně ten nejjednodušší kód vypadá takto:

```
<canvas width="300" height="225"></canvas>
```

Na stránce můžete mít kolik canvasů chcete. Když mu přidělíte atribud ID, tak k němu můžete dále přistupovat jako ke každému jinému elementu a živě ho upravovat. Třeba javascriptem.

```
<canvas id="a" width="300" height="225"></canvas>
```

Tak, nyní můžeme s canvasovým polem dále pracovat, třeba přes javascript. V DOMu ho můžete snadno najít tímto zápisem:

```
var a_canvas = document.getElementById("a");
```

První ukázka

Každé plátno je na začátku prázdné. A to je nuda! Pojďme něco nakreslit.

V první živé ukázce vytváříme plátno o velikosti 300x225px a nastavíme mu rámeček. Následně ho pojmenujeme ID=“b“ a vytvoříme mu událost onclick.
Někam pod plátno pak stačí vložit odkaz, který po aktivaci spustí javascriptovou část, která řekne canvasu, aby něco udělal.
Když to co nejvíce smrsknu, tak to může vypadat třeba takto:

```
<script type="text/javascript">
function draw_b() {
  var b_canvas = document.getElementById("b");
  var b_context = b_canvas.getContext("2d");
  b_context.fillRect(50, 25, 150, 100);
}
</script>
<canvas id="b" width="300" height="225" style="border:1px dotted;" onclick="draw_b();return false"></canvas>
<a href="#" onclick="draw_b();return false">aa</a>
```

Vyzkoušejte si to. Můžete přidat i další objekty a nakreslit i o mnoho složitější obrazce. Stačí jen psát další a další kreslící kódy do javascriptu v horní části stránky, přesně před závorku (zvýrazněno).

Otázka z webového fóra:

Otázka: Existuje i 3D canvas?
Odpověď: Zatím ne. Někteří výrobci prohlížečů experimetovali s vlastními API pro 3D canvas, ale žádné zatím nebylo standardizováno. Specifikace HTML5 zmiňuje, že „budoucí verze této specifikace pravděpodobně budou definovat i 3D prostředí“.

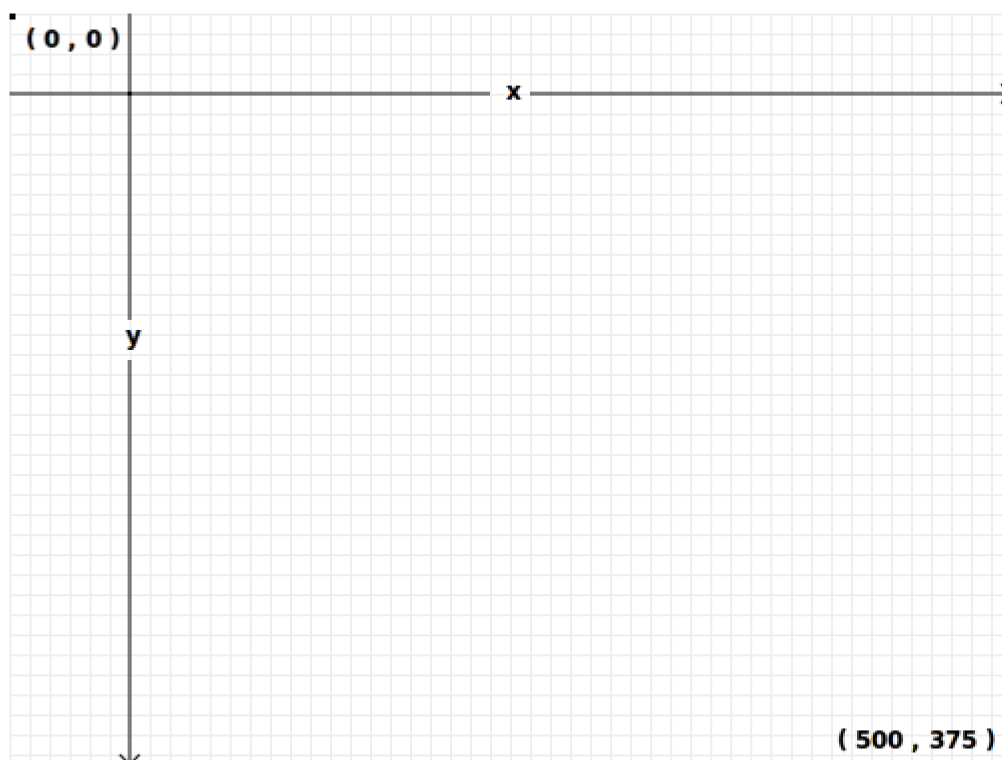
Takže, máme element <canvas> a máme jeho kreslicí prostředí. V tomto prostředí jsou definovány všechny vlastnosti a metody kreslení. Kreslení obdélníků se věnuje celá skupina vlastností a metod:

- Vlastnost fillStyle (styl výplně) může být CSS barva, vzorek nebo přechod. (Více o přechodech později.) Implicitně je fillStyle nastavena jako černá, ale můžete ji nastavit jakkoliv chcete. Každé kreslicí prostředí si pamatuje své vlastnosti, dokud stránku nezavřete nebo je sami nezresetujete.
- fillRect(x, y, šířka, výška) nakreslí obdélník s aktuálním stylem výplně.
- Vlastnost strokeStyle (styl orámování) je podobná vlastnosti fillStyle — může být CSS barvou, vzorkem nebo přechodem.
- strokeRect(x, y, šířka, výška) nakreslí obdélník s aktuálním stylem orámování. Vlastnost strokeRect nevyplňuje střed, nakreslí pouze okraje.
- clearRect(x, y, šířka, výška) odstraní pixely v definovaném obdélníku.

Plátno ještě jednou

Plátno si představte jako soustavu souřadnic v geometrii. Je to vlastně dvojrozměrná mřížka. Souřadnice (0,0) jsou levý horní rok plátna a vše se odtud vykresluje až na maximální možné souřadnice, které si určíte za pomoci width a height přímo v HTML stránce.

Plátno si můžete představit zhruba takto:

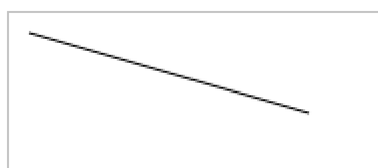


Toto je sice jen prostý obrázek, ale dal by se také canvasem nakreslit. Nebudu zde uvádět konkrétní kód (ten si každý bude umět napsat sám podle předchozího příkladu a funkcí, které v knize naleznete). Toto je jen teoretické kreslení, ale dalo by se to udělat zhruba takovýmto způsobem:

- soustavy světle šedých vertikálních čar
- soustavy světle šedých horizontálních čar
- dvě černé horizontální čáry
- dvě malé čáry, které vytváří šipku
- dvě černé vertikální čáry
- dvě malé čáry, které vytváří druhou šipku
- písmenko „x“
- písmenko „y“
- text „(0, 0)“ poblíž levého horního rohu
- text „(500, 375)“ u pravého spodního rohu
- tečka v pravém horním rohu a druhá v levém spodním rohu

Kreslení čar

Není to vůbec složité. Jedná se vlastně o funkci, která zavede kreslení, pak funkci která určí odkud kam povede a nakonec samotné vykreslení.



Jednoduchou čáru můžeme udělat třeba takto:

```
ctx.moveTo(10,10);
ctx.lineTo(150,50);
ctx.stroke();
```

Tentokrát se javacriptová část píše až za objekt <canvas>, obráceně to nefunguje. Je to proto, že nejprve vytvoříme prázdné kreslící plátno a až později do něj dokreslujeme jednotlivé objekty dle libosti za pomocí javascriptu. Celé to vypadá zhruba takto:

```
<canvas id="kreslenicary" width="300" height="150" style="border:1px solid“></canvas>
<script type="text/javascript">
var c=document.getElementById("kreslenicary");
var ctx=c.getContext("2d");
ctx.moveTo(10,10);
ctx.lineTo(150,50);
ctx.stroke();
</script>
```

Kde toho najdu o Canvasu více?

Toto je celý začátečnický canvasový tutoriál. K canvasu se budu postupně v knize ještě vracet a hojně ho využívat. Těchto prvních pár stran vám mělo přiblížit, jak canvas vlastně funguje a už by jste měli být schopni vytvořit vlastní jednoduchý obrázek. Některé dobré online tutoriály:
https://developer.mozilla.org/en/Canvas_tutorial (tutoriál od Mozilly, anglicky)
http://www.canvasdemos.com/ (dema, nástroje a tutoriály k HTML elementu canvas)
http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html
http://msdn.microsoft.com/en-us/ie/ff468705.aspx#_HTML5_canvas (canvas od Microsoftu)

FORMULÁŘE

Toto téma je opravdu obsáhlé. V HTML5 přibylo spoustu zajímavých formulářových řešení, které mohou hodně usnadnit ovládání vašeho webu.

Typ	Využití
Datetime	Výběr datamu a času
Datetime-local	Výběr datamu a času
Date	Výběr datumu
Month	Výběr měsíce
Week	Výběr týdnu
Time	Výběr času
Number	Pole pro číslo
Range	Posouvač, výstupem je číslo (min-max)
Email	Pole pro email + kontrola formátu
Url	Pole pro URL
Search	Pole pro vyhledávání
Color	Pole pro výběr barvy + textový formát

Formuláře se zapisují stejně jako v HTML4. Jednoduchý příklad:

<input type=“date“ name=“datum“>

Stačí jen hodnotu z tabulky napsat do atributu type.

TIP: Všechny formuláře si vyzkoušejte v Opeře. Zde je v současné době nejlepší podpora a vykreslení.

Na následující straně začíná dlouhá tabulka. Bude v ní vždy obrázek, jak se formulář zobrazuje, (případně ukázku kódu), nějaké specifikace a nebo případný popis.

DATETIME

Pole sloužící pro vybrání datumu a času. Když formulář odešlete metodou GET, tak můžete dostat např. Takovoutu adresu:

http://baraja.cz/formular.html?t=2012-04-12T12%3A00Z

DATE

Výběr datumu. Posílá např.: 2012-04-12

```
<form>
  <input name="datum" type="date">
  <input type="submit" value="Odeslat">
</form>
```

MONTH

Vybírá celý měsíc

WEEK

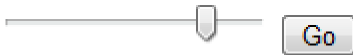
Výběr týdne

TIME

U formuláře vytvoří šipky pro možnost posouvání. Lze upravit i přímím vložením hodnot na klávesnici.

NUMBER

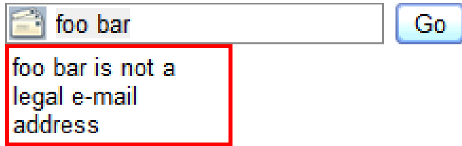
Umožňuje vybrat konkrétní číslo. Přidává šipky na posouvání.



RANGE

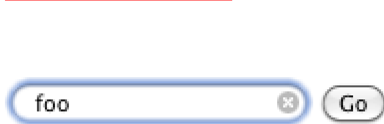
Vytvoří posuvník. Vybírá od minimální po maximální nastavenou hodnotu.

<input name="r" type="range" min="1" max="11" value="9">



EMAIL
Pole pro zadání emailu + kontrola formátu (zavináč)

URL
Pole pro URL adresu
SEARCH

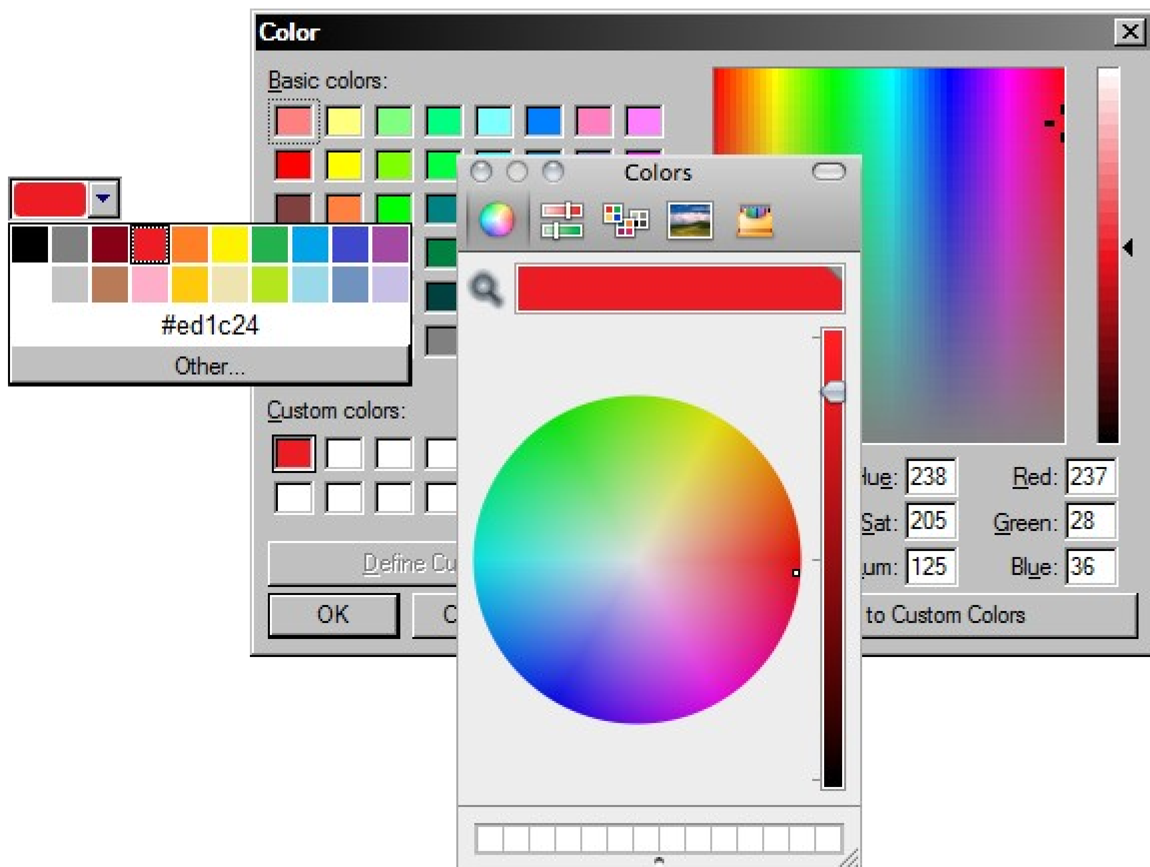



```
<form>
  <input type="search">
  <input type="submit" \
</form>
```

Vyhledávací políčko.
Jedné co umí je, že vkládá na pravo do políčka křížek pro smazání.

Color (Vybrání barvy)

Základní formulářové políčko (v levo). Uprostřed podrobnější výběr barvy ve Windows a uprostřed je přesnější vybrání barvy v Macu.



Kontrola validity zadaných údajů

Každému vývojářovi se jistě hodí kontrolovat správnost vyplnění formulářových políček ještě před odesláním. Má to několik výhod:

- Klient nemusí čekat na zpracování serverem
- Nemusí se načítat další stránka
- Políčko lze snadno opravit
- Vypadá to dobře
- Je to jednoduše proveditelné

Jednoduchý příklad:

```
<form>
  <input name="email" placeholder="Zadejte váš email">
  <input type="submit" value="Odeslat">
</form>
```

Toto ještě není žádná složitá interakce, ale prostý šedý text vložený do políčka. Zmizí jakmile něco napíšete.

Obyčejný formulář

Vylepšený formulář

Search Bookmarks and History

Interakce poprvé

Ještě nedávno tohle všechny bylo poměrně složitě proveditelné. Muselo se to složitě řešit přes javascript, bylo to trochu náročnější a málokdo se v tom vyznal. HTML5 se toto všechno snaží napravit a všechny nejčastější funkce vkládá do pár tagů. Tato interakce spočívá v opravě správně napsaných údajů do formulářových políček. Složitější funkce, jako třeba zkontrolování zda zadané uživatelské jméno ještě existuje si musíte stále udělat sami a ověřit přes Ajax (což málokdo zvládne sám).

V HTML5 se o to stará atribut required. Ničím se dále nerozvíjí, ale napíše se jen jako slovo. Příklad:

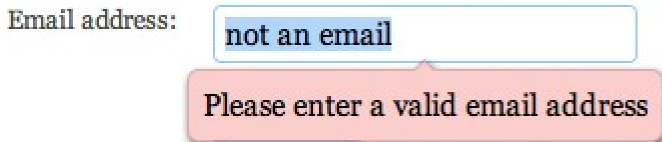
```
<form>
   <input name="policko" required>
   <input type="submit" value="Odeslat“>
</form>
```

Tento zápis vytvoří formulářové políčko a tlačítko k odeslání. Když kliknete na odeslat a nebude nic vyplněno, tak vás to dále nepustí a vyhodí to hlášku, že máte pole doplnit.

Interakce podruhé

To samé můžete udělat i pro formulář, kde se píše email a podobně. Zde to bude kontrolovat výskyt zavináče, doménového jména, tečky a přípony. V případě neúspěchu se vyhodí opět hláška.
Je celkem škoda, že si nemůžete sami určit, co bude taková hláška obsahovat za přesný text, ale i tak to je dobré. Jazyk hlášky, vzhled a podobně si dělá prohlížeč plně sám.

```
<form>
   <input type="email" name="policko" required>
   <input type="submit" value="Odeslat">
</form>
```



To samé můžete udělat i s ostatními formulářovými políčky. Více najdete na stránkách Mozilly:
http://blog.mozilla.org/webdev/2011/03/14/html5-form-validation-on-sumo/

Samozřejmě nezapomeňte formulář ještě ošetřit za pomoci PHP nebo jiného serverového jazyku. Nemůžete se totiž na prohlížeč plně spolehnout. Jak to udělat se dočtete v části knihy, kde se zaobírám PHP a jeho možnostmi.

Interakce potřetí

Toto je velice hodně zjednodušená ukázka toho, jak v reálném čase pracovat s čísly a formulářem. Na stránce mám posuvník, který určuje rozmezí čísel 0 až 100 a vedle speciální políčko, které hodnotu okamžitě vypisuje.

71

```
<form oninput="x.value=parseInt(a.value)">
<input type="range" name="a" value="50">
<output name="x" for="a"></output>
</form>
```

Toto je velice zjednodušená ukázka. Všimněte se barevně zvýrazněných částí kódu. Nahoře v části <form> se děje celá magie. Tam se nastartuje proměnná **x**, která přenáší hodnotu.
Vlastně přijme proměnnou **a** z formuláře a překopíruje jí na proměnnou **x**, která se vypíše ve formuláři <output>

Nebojte se experimentovat a spojit to více s javascriptem. Jednou můžete udělat přes takovouto interakci plno zajímavých věcí.

Vlastní interakce

Atribut required se dá dále ještě lehce poupravit a můžete mu dát novou funkci. V základu bude vždy kontrolovat zda není formulář prázdný, ale vy mu můžete přidat ještě o podmínku navíc.
Uvedu příklad: Máte formulář pro vložení telefonního čísla. Tak je dobré vložit podmínku, že číselný výraz musí mít právě 9 číselných znaků. To uděláte takto:

```
<form>
<input type="text“ name="telefonni-cislo" required="required" pattern="[0-9]{3}[0-9]{3}[0-9]{3}"
title="Musíte zadat číslo dlouhé 9 znaků!“>
<input type=“submit“ value=“Odeslat“>
</form>
```

Shrnutí formulářů

Na celkový přehled o tom, jak si vede jaký prvek jsem připravil obsáhlou tabulku, vše otestoval a takto to dopadlo:

Typ	Opera 11.10	Google chrome 11	Firefox 4	Internet explorer 9
datetime	Výborně	Lehce nepřehledné	Nepodporuje	Nepodporuje
date	Výborně	Výborně	Nepodporuje	Nepodporuje
month	Výborně	Výborně	Nepodporuje	Nepodporuje
week	Výborně	Divný zápis	Nepodporuje	Nepodporuje
time	Výborně + za posuvníky	Výborně	Nepodporuje	Nepodporuje
number	Výborně	Výborně	Nepodporuje	Nepodporuje
range	Výborně + za oddělovače	Výborně	Nepodporuje	Nepodporuje
email	Nepodporuje	Nepodporuje	Výborně + za zpětnou vazbu	Nepodporuje
url	Dobrý + za doplnění http://	Nepodporuje	Nepodporuje	Nepodporuje
search	Nepodporuje	Výborně + za křížek	Nepodporuje	Nepodporuje
color	Výborně + za paletu	Nepodporuje	Nepodporuje	Nepodporuje

Nové vlastnosti:

Atribut	Popisek
placeholder	Zobrazí předvyplněný text, který po kliknutí zmizí a jde do inputu psát
multiple	Povoluje vybrat více souborů (Držte CTRL a vybírejte...)
required	Kontrola údajů ve formuláři. Zkuste něco zadat a stiskněte enter. Pokud podmínky dodržíte, formulář se odešle. V opačném případě se ukáže hláška.
autocomplete	Samovyplnění prohlížečem, nastavuje se: ON/OFF

VIDEO A AUDIO

Když jste chtěli v HTML4 vložit do stránky video, tak jste se neobešli bez nějakého doplňku. Třeba flash přehrávače a nebo toho výmyslu od YouTube. HTML5 se toto snaží napravit a nyní můžete ve stránce přehrávat video rovnou.
Do tohoto kroku se zatím příliš nepouštějte, protože zatím ještě neexistuje žádný standart, který by říkal, jaký má video mít formát a jaký kodek zvuku.

Podpora formátů:

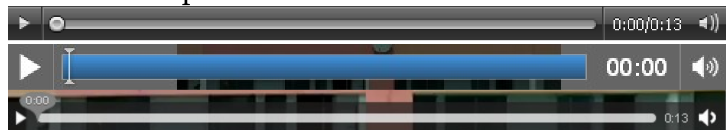
Formát	Opera 11.10	Google chrome 11	Firefox 4
WEBM	1/1	1/1	1/1
AVI	0/0	0/0	0/0
MP4	0/0	1/1	0/0
OGG	0/1	0/1	0/1

Údaje v tabulce jsou psány stylem: 1 = podporuje, 0 = nepodporuje. První údaj je obraz, druhý zvuk.

Jaký formát využijete je čistě na vás. Můj názor je, že se HTML5 video pro masivní nasazení ještě stále nehodí. Kód:

```
<video src="video.webm" controls="controls">
Toto se zobrazí, pokud prohlížeč HTML5 videa nepodporuje
</video>
```

Zobrazení v prohlížečích:



Opera
Chrome
Firefox

Audio má zápis podobný:

```
<audio src="pisnicka.webm" controls="controls">
Toto se zobrazí, pokud prohlížeč HTML5 přehrávač na písničky nepodporuje
</audio>
```

Výhoda HTML5 přehrávače netkví jen v tom, že je nenáročný, ale také v tom, že si ho můžete libovolně oskynovat za pomoci CSS a nebo si dát ovládací tlačítka na místa, kde se vám to zrovna hodí. Na to se dají použít tyto skvělé funkce: AddTextTrack(), canPlayType(), load(), play(), pause()

více se o HTML5 videu dočtete na této stránce:
http://www.w3schools.com/html5/html5_ref_av_dom.asp

STRUKTURA

Je každá stránka stejná? Ne, není. Ale jednotlivé části ano. Proč psát stále dokola to samé, když to můžeme zkrátit na jednoslovný zápis? Přesně toto HTML5 začíná řešit.
Kvůli tomuto byly v HTML5 zavedeny nové tagy.

- header - reprezentuje hlavičku stránky
- nav - reprezentuje část stránky, která je určena k navigaci, nejčastěji menu
- article - reprezentuje nezávislé textové části stránky, např. komentáře, články,
- section - reprezentuje různé části stránek, např. kapitoly
- aside - reprezentuje boční panel stránky, který se moc neváže k obsahu hlavní stránky
- footer - reprezentuje patičku stránky, může obsahovat informace o autorovi, autorských právech nebo odkazy na související dokumenty
- figure - reprezentuje část stránky s informacemi, které k sobě patří, např. video a jeho popisek

Ukázka:

```
<body>
   <header></header>
   <nav></nav>
   <aside></aside>
   <section>
      <article></article>
   </section>
   <footer></footer>
</body>
```

Pak se to v reálné situaci řeší tak, že si v CSS nastavíte, jak bude každý takovýto objekt vypadat a ušetříte si spoustu řádků kódu.



OFFLINE

Počkat? WTF!!! Oni jsou dnes ještě lidé, co nemají přístup k internetu?
HTML5 přichází s celkem revoluční funkcí, kdy si dopředu určíte, které části aplikace se mají stáhnout na počítač klienta a v případně nedostupnosti připojení stačí vyťukat stejnou adresu do adresního řádku a aplikace se spustí plně offline. Jakmile se uživatel opět připojí, tak se data synchronizují se serverem a vše se vrátí do stavu online. Toto používá např. Google u svých dokumentů, Gmailu, chatu a podobně. Myslím že je celkem příjemná věc, že po otevření stránky s emaily se mi prvních 20 stáhne do cache prohlížeče a v případě nečekaného pádu můžu číst dál. Dokonce i odpovědět a ihned jak se připojím se vše odešle.
V minulosti byl toto problém. Jediný způsob jak ukládat na stanici klienta bylo cookies. Bylo to dost nepraktické, pomalé, vešlo se tam jen 4KB a musely to být textové informace, nikoliv celá stránka nebo aplikace.

TIP: Než se do toho pustíte, tak si zkuste přes knihovnu Modernizr (modernizr.com) ověřit, zda je lokální uložení dostupné. Pokud ano, tak můžete uživatele potěšit nějakým příjemným textem, že se nemusí na vašem webu surfovat i v režimu offline a pokud ne, tak ho upozorněte, že by si měl dávat na své připojení bacha nebo stáhnout lepší prohlížeč.

```
if (Modernizr.localstorage) {
  // Gratuluji, můžete ukládat na lokální úložiště!
} else {
  // Bohužel, lokálně ukládat nejde.
}
```

Dejte si ale pozor na to, co chcete všechno offline ukládat a je vhodné o tom nejprve uživatele informovat. Každý kdo má totiž přístup k počítači, tak si může data prohlížet v cache a nebo je dokonce měnit!

Každá offline aplikace vždy začíná jako obyčejná online aplikace na internetu. Uživatel stránku otevře a stránka prohlížeči řekne co všechno bude potřebovat. Prohlížeč si postupně všechno postahuje (může se jednat o obyčejné texty, html, javascript, obrázky, videa, cokoliv).
Jakmile se všechno stáhne a vy stránku znovu načtete offline, tak by jste ani neměli poznat, že offline skutečně jste.

Ukázka první

Tento kód se snaží zjistit, zda jste online nebo offline. Když není HTML5 podporováno, tak se vypíše, že to nelze zjistit.

```
<!DOCTYPE HTML>
<html>
 <head>
  <title>Kontrola dostupnosti</title>
  <script>
   function updateIndicator() {
     document.getElementById('indicator').textContent = navigator.onLine ? 'online' : 'offline';
   }
  </script>
 </head>
 <body onload="updateIndicator()" ononline="updateIndicator()" onoffline="updateIndicator()">
  <p>Právě jste: <span id="indicator">(nelze zjistit)</span>
 </body>
</html>
```

Ukázka druhá

Neříkejte, že by se vám nehodila rada jak uložit nějaké data do cache prohlížeče. Ono to vlastně není v prohlížeči, ale budeme dělat jako že ano.
V této ukázce si nemá cenu hrát na nějakou velkou aplikaci, a proto udělám jen jednu stránku s nějakým prvkem. Úplně na začátku (při první návštěvě) je nutné prohlížeči správně určit, které soubory bude potřebovat. Později, když nebudete připojeni k internetu se pak prohlížeč podívá právě na ty soubory, které mu určíte a s těmi bude pracovat.
Udělat takto jednoduchou statickou stránku typu blog je relativně jednoduché, horší to je, pokud chcete udělat aplikaci která má mít nějakou interakci. Například formulářové políčko pro vaše osobní poznámky. Tam se to řeší tak, že si prohlížeč stáhne stránku s formulářem a nějaký obslužný javascript. A když bude zrovna k dispozici připojení, tak se data přes ajax odešlou na server.
Všechny soubory (data), které budete v offline režimu potřebovat se ukládají do místa zvané „.appcache“. Nyní si tedy určíme, co vše budeme potřebovat:

```
CACHE MANIFEST
#tento soubor se stáhne do režimu offline
 offline.html
NETWORK:
#Tyto soubory se neukládají a jsou použity jen v online režimu
 statistika.gif
CACHE:
#Zde jsou soubory, které nejsou v on-line režimu použity, přesto se uloží do cache
 offline.html
 kocka.png
FALLBACK:
offline.html
```

Důležité pojmenování souborů

Jak je vidět, lze použít i poznámky. Musejí být ale na samostatném řádku a před nimi znak mřížky (#).

Ještě pro upřesnění: Tento kód, který určuje co všechno se má stáhnout pojmenujte „**.appcache**“. Ano, soubor se bude jmenovat **tečka appcache**!

Dále bude potřeba vytvořit ještě jeden soubor, který bude mít úplně stejný obsah a bude se jmenovat: „application.appcache“. Proč to musí být 2x, to těžko říct. Zkoušel jsem to smrštit do jednoho, ale nějak se mi to nedařilo.

A hurá na aplikaci!

Úplně první aplikace bude vlastně úplně obyčejná HTML stránka co bude mít jeden obrázek a krátký text. Nesmíte na začátku dokumentu zapomenout uvést, že si přejete soubory ke klientovi stáhnout a dát tak prohlížeči najevo, že se mu to bude vážně hodit.
Zbytek je pak už klasický HTML kód, případně javascript. Můžete si se stránkou vyhrát naprosto jak chcete.

```
<!DOCTYPE html>
<html manifest="application.appcache">
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8“>
        <title>Testovací offline webová aplikace</title>
</head>

<body>
  <h1>Nadpis stránky</h1>
  Toto je zkušební offline aplikace. Schválně, odpojte se od internetu a zkuste znovu načíst stránku!
  <br><br><br>
  <img src="kocka.png">
</body>
</html>
```

Nefunguje mi to... ;(

To je možné. Mě to fungovalo jen v prohlížeči **Google Chrome**. V době psaní tohoto článku zatím nemá cenu počítat s tím, že prohlížeč klienta bude toto umět, ale můžete se o to pokusit.

Další nejčastější problém je špatně nastavený server. Koukněte se, jestli máte vytvořený soubor .htaccess a jestli v něm je tento řádek kódu:

```
AddType text/cache-manifest .appcache
```

Je to vážně důležité, jestli si chcete stránky prohlížet i jinde než na internetu.
Další, a to málo častá chyba je tak, že se webová stránka nestihla celá stáhnout. Myslete také na to, že klient nemusí mít rychlé připojení. Proto je dobré nějak přes javascript otestovat, zda se všechny potřebné soubory stáhly. Nejprve tedy stáhněte HTML stránku (tu nejzákladnější co v případě nedokončení stažení napíše chybu) a až pak zbytek.

Ukázka třetí

Jistě se vám může hodit přes javascript vytáhnout aktuální stav. Můžete tedy velice snadno zjistit, jestli se třeba právě stahují data do cache, jestli je už všechno staženo a nebo jestli je už připraveno vše na update (odeslání offline vytvořených dat na server).

Slouží k tomu objekt, který se jmenuje ApplicationCache.

Má jednu vlastnost: status (pozor, je jenom ke čtení). Posílá aktuální stav pro aktuální dokument. Stav posílá formou čísla. Vše jsem shrnul do přehledné tabulky:

0	UNCACHED	Stránka nemá přiřazen manifest, nebo ještě nebyla načtena
1	IDLE	Aplikační cache je aktuální a kompletní
2	CHECKING	Prohlížeč posuzuje aktuálnost manifestu
3	DOWNLOADING	Stahování nového obsahu do aplikační cache
4	UPDATEREADY	Připraveno pro update
5	OBSOLETE	Manifest nenalezen, aplikační cache bude vyprázdněna

Ještě se vám můžou hodit další funkce, třeba na aktualizaci právě načtených dat a podobně. K čemu je vlastně funkce na aktualizaci, když je v prohlížeči tlačítko refresh? Ať se to zdá jako vytěžovačka připojení, tak spíše naopak to pomáhá. Když třeba chcete nechat uživatelům stáhnout nejnovější emaily a nebo nejnovější články z vašeho zpravodajského serveru, tak je tato funkce přímo pro vás.

Jmenuje se: Update()

Funguje to tak, že si stáhne nové (aktuální) data z internetu do cache. Pozor, jen do cache, v prohlížeči stále uvidíte tu starší verzi. Proto musíte ještě vyvolat funkci swapCache(), která data obnoví.

Myslete na starší prohlížeče

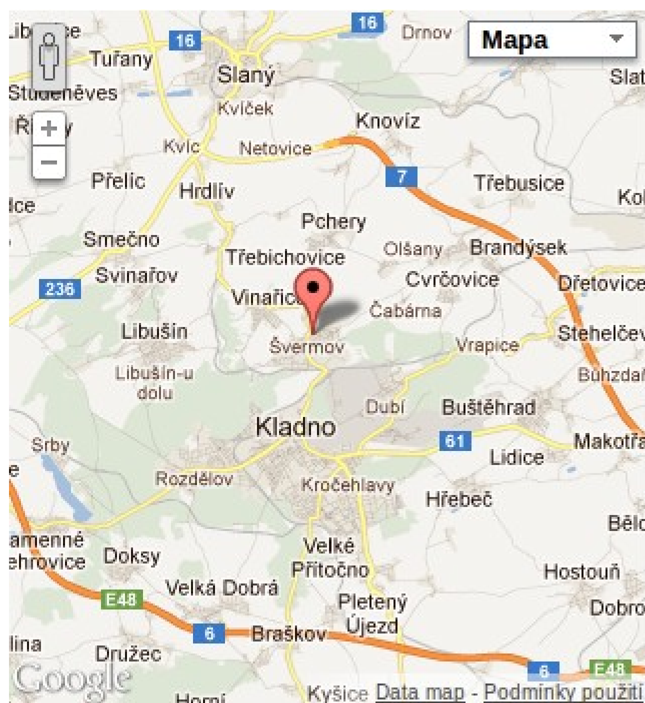
Starší či zastaralejší prohlížeče nemusí cachování přes HTML5 plně podporovat, tak se celkem hodí jim pomoci a ošetřit to objektem ApplicationCache a dát přes to podmínku.

```
if (window.ApplicationCache && window.WebSocket) {
   var server = new WebSocket("http://example.com/manifest/");
   server.onmessage = function() {
      window.applicationCache.update();
   };
   window.applicationCache.addEventListener("updateready", function() {
      window.applicationCache.swapCache();
   }, false);
}
```

Po zkušenostech vím, že to u různých prohlížečů může pomoci nastartovat dočasné úložiště a jejich majitelům zásadně zpříjemnit život.

GEOLOKACE

Tohle neocení jen „kačeři“, ale i všichni ostatní. Jestli máte stránky vaší firmy, tak se může i celkem hodit vložení speciálního doplňku, mapy a zvýraznit cestu k vám. Další využití mě napadá třeba u mapových portálů a nebo třeba lokálního hledání. Co mě vždycky potěší na stránce je to, že po otevření nějakého katalogu (třeba firem) mi přednostně ukáže ty v okolí a až pak ty další.



Zhruba takováto mapka se mi zobrazila, když jsem si vyzkoušel lokalizační script. Úmyslně jsem to oddálil (nikdo nemusíte vědět kde právě jsem), ale po přiblížení je reálná odchylka zhruba 10 metrů a to nemám v notebooku žádné zařízení GPS ani nic podobného.

Funkce to je poměrně zajímavá, ale ať nikoho nenapadne takhle zaměřovat vaše uživatele a někam si to ukládat, mohlo by se jim to také nelíbit a nebo by vás mohli obvinit z neoprávněného nakládání s osobními údaji.

Dobrá, strašit už nebudu. Každý si to přeberte jak chcete a vrhněme se na to.

Na toto nemá cenu vymýšlet vlastní řešení, když vám Google nabídne právě takto přesní zaměření, použijme tedy jeho API. Ukázka celé stránky:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset=utf-8>
<meta name="viewport" content="width=620">
<title>Hledání na schovávanou</title>
<link rel="stylesheet" href="css/html5demos.css">
<script src="http://html5demos.com/js/h5utils.js"></script></head>
<body>
<section id="wrapper">
<div id="carbonads-container"><div class="carbonad"><div id="azcarbon"></div><script
type="text/javascript">var z = document.createElement("script"); z.type = "text/javascript";
z.async = true; z.src = "http://engine.carbonads.com/z/14060/azcarbon_2_1_0_VERT"; var s =
document.getElementsByTagName("script")[0]; s.parentNode.insertBefore(z,
s);</script></div></div>
```

```
  <header>
    <h1>Hra na schovávanou</h1>
  </header>
<meta name="viewport" content="width=620“>

<script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>
  <article>
    <p>Právě jsi tady: <span id="status">Hledání...</span></p>
  </article>
<script>
function success(position) {
  var s = document.querySelector('#status');

  if (s.className == 'success') {
    return;
  }

  s.innerHTML = "found you!";
  s.className = 'success';

  var mapcanvas = document.createElement('div');
  mapcanvas.id = 'mapcanvas';
  mapcanvas.style.height = '400px';
  mapcanvas.style.width = '560px';

  document.querySelector('article').appendChild(mapcanvas);

  var latlng = new google.maps.LatLng(position.coords.latitude, position.coords.longitude);
  var myOptions = {
    zoom: 15,
    center: latlng,
    mapTypeControl: false,
    navigationControlOptions: {style: google.maps.NavigationControlStyle.SMALL},
    mapTypeId: google.maps.MapTypeId.ROADMAP
  };
  var map = new google.maps.Map(document.getElementById("mapcanvas"), myOptions);

  var marker = new google.maps.Marker({
      position: latlng,
      map: map,
      title:"Mám tě! (at least within a "+position.coords.accuracy+" meter radius)"
  });
}

function error(msg) {
  var s = document.querySelector('#status');
```

```
  s.innerHTML = typeof msg == 'string' ? msg : "failed";
  s.className = 'fail';

  // console.log(arguments);
}
if (navigator.geolocation) {
  navigator.geolocation.getCurrentPosition(success, error);
} else {
  error('not supported');
}

</script>
<script src="http://html5demos.com/js/prettify.packed.js"></script>
<script>
var gaJsHost = (("https:" == document.location.protocol) ? "https://ssl." : "http://www.");
document.write(unescape("%3Cscript src='" + gaJsHost + "google-analytics.com/ga.js'
type='text/javascript'%3E%3C/script%3E"));
</script>
<script>
try {
var pageTracker = _gat._getTracker("UA-1656750-18");
pageTracker._trackPageview();
} catch(err) {}</script>
</body>
</html>
```

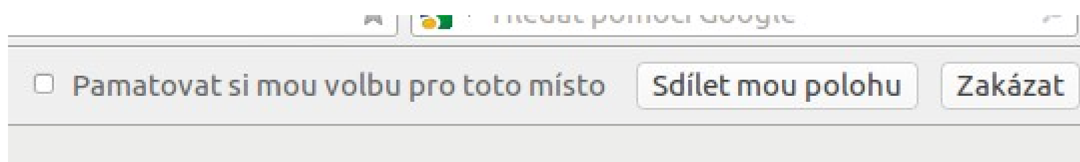
Tento příklad byl převzat ze stránek http://html5demos.com/, můžete si to online vyzkoušet zde: http://html5demos.com/geo

A jak to ve zkratce funguje? Všechno to záleží na objektu navigator.geolocation, který se o vše stará. Můžeme opět vyzkoušet podporu:

```
if (navigator.geolocation) {
alert("Tak, a jde se zaměřovat!");
} else {
alert("Škoda, tvůj prohlížeč to neumí.");
}
```

A co povolení?

Ano, prohlížeč se vás při každém pokusu o lokalizaci dotáže. Nebo alespoň v továrním nastavení. Když mu povolení nedáte, tak jste ztraceni.



SVG A OBRÁZKY

Když se řekne „automatické kreslení obrázků v HTML5“, tak si většina lidí představí Canvas. Canvasu jsem na začátku knihy věnoval několik stránek, nyní si ale představíme další řešení: SVG.
On je to vlastně normální formát pro obrázky. Není problém mít obrázky v počítači uložené ve formátu *.svg. HTML5 to ale dotáhlo ještě dál.
Když můžete mít obrázky uložené v počítači jako soubory, proč by je nešlo generovat přímo v prohlížeči? Všechno jde! Další výhoda od Canvasu je podpora, jelikož se to zobrazí téměř všude.

Jak to funguje?

Snadněji než si myslíte! Oproti Canvasu mě nejvíce překvapilo tak snadné řešení a zápis, že jsem nemohl uvěřit, že to skutečně funguje. Fungovalo to.
Všechno to začíná prvním tagem, který určí že se jedná o SVG obrázek a že to je verze 1.1

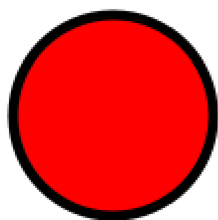
```
<svg version="1.1">
```

Dále je už samotný zápis a nakonec jen ukončení párového tagu:

```
</svg>
```

Když to celé zkrátím, tak to vypadá zhruba takto:

Moje první kružnice



```
<html>
<body>
<h1>Moje první kružnice</h1>
 <svg version="1.1">
   <circle cx="100" cy="60" r="50" stroke="black" stroke-width="5" fill="red">
 </svg>
</body>
</html>
```

Další obrovská výhoda SVG je, že to je vlastně XML. Zápis je velmi podobný, vlastně úplně stejný. Pokud už umíte pracovat s XML, tak přestup na SVG obrázky pro vás nebude vůbec obtížné. Naopak. Získáte spoustu zajímavých možností a vaše obrázky na webu získají užitečnou vlastnost: Půjdou libovolně zoomovat (přibližovat) bez ztráty kvality. Zkrátka to je jednoduše vygenerovaná vektorová grafika. Na příklad Wikipedie má většinu obrázků právě v SVG a snaží se takto všechny převést, jelikož jí to dává možnost si s obrázek libovolně hrát, libovolně deformovat, zvětšovat, zmenšovat a nezničit tak jeho kvalitu a rast, který vlastně nemá žádný.
SVG zápis vlastně neobsahuje žádný obrázek, ale jen všechny potřebné informace k jeho vytvoření. Obrázek tedy vlastně vůbec „fyzicky“ neexistuje, ale pokaždé se znovu vygeneruje. To má obrovskou výhodu pro kapacitu a ztrátu kvality.

Možnosti vložení

Jsou celkem 4 způsoby:

- Jako zdrojový kód (ukázka na předchozí straně)
- Jako klasický obrázek
- Jako rámec (iframe)
- Jako odkaz

Pokud máte obrázek uložený v samostatném souboru, tak je zápis poměrně snadný:

<embed src="obrazek.svg" type="image/**svg+xml**">

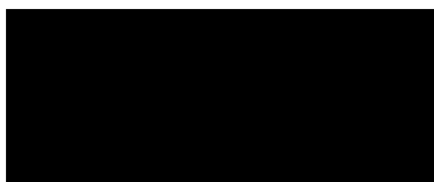
Hodí se i nastavit **type** na **svg+xml**, což prohlížeči řekne, že se nejedná o klasický obrázek, ale pouze o jeho zdrojový kód v jazyce XML a je nutné ho vygenerovat.

Jako rámec
To se dá udělat zhruba takto. Ale pozor, nemusí se to vždy zobrazit korektně

<iframe src="circle1.svg"></iframe>

Ukázka první

Je vůbec něco jednoduššího než obdélník přes SVG?



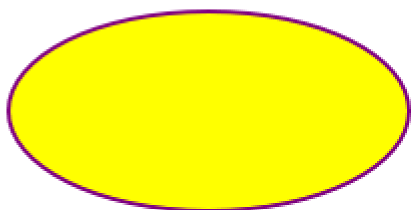
```
<svg version="1.1">
<rect width="150" height="75">
</svg>
```



```
<svg version="1.1">
<rect width="150" height="75" style="fill:rgb(253, 171, 7);">
</svg>
```

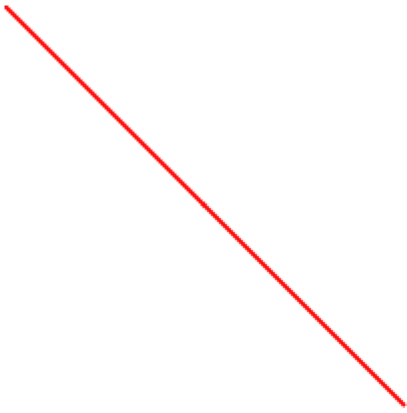
Ukázka druhá

Aneb, jednoduše na elipsy:



```
<svg version="1.1">
<ellipse cx="300" cy="80" rx="100" ry="50"
  style="fill:yellow;stroke:purple;stroke-width:2"/>
 </svg>
```

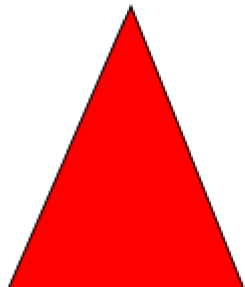
Ukázka třetí



Čáru uděláte také velice snadno. Jen určíte počáteční a konečné body. Je tu snad někdo, kdo to nechápe?

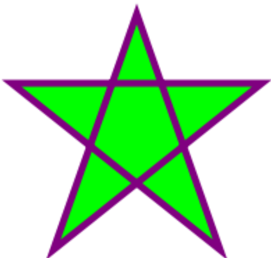
```
<svg version="1.1">
<line x1="0" y1="0" x2="200" y2="200"
  style="stroke:rgb(255,0,0);stroke-width:2"/>
 </svg>
```

Ukázka čtvrtá



Aneb, víceúhelník. Stačí jen psát souřadnice míst, kudy chcete táhnout jednotlivé body. Minimálně jsou 2 místa a maximální počet není teoreticky omezen.

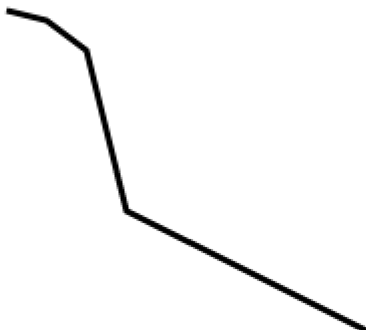
```
<svg version="1.1">
<polygon points="220,10 300,210 170,250 123,234“
style="fill:red;stroke:black;stroke-width:1">
 </svg>
```



Můžete udělat i komplikovanější obrázky. Například hvězdu:

```
<svg version="1.1">
<polygon points="100,10 40,180 190,60 10,60 160,180"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:nonzero;" />
 </svg>
```

Ukázka pátá



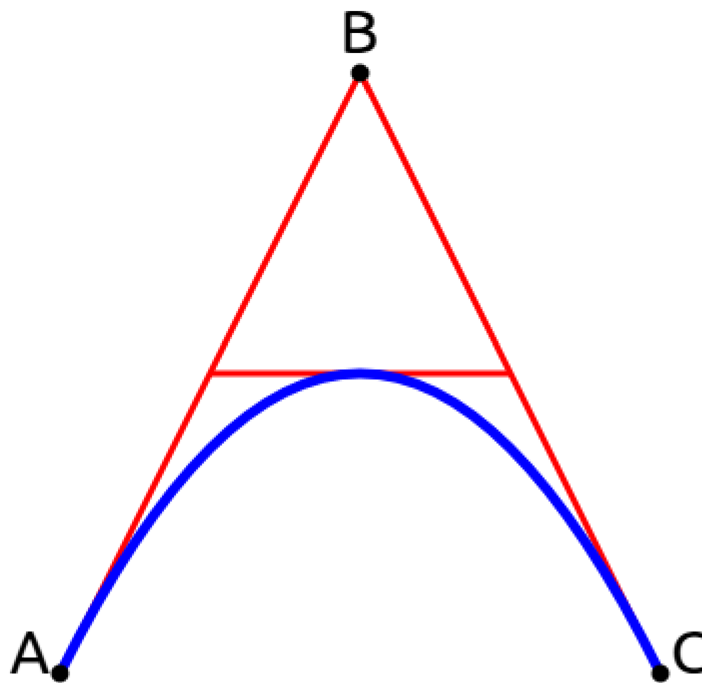
Tato čáranice je vlastně čára s více body. Toto konkrétně se dá vyřešit tak, že vložíte více čar za sebe, ale dá se to i zkrátit:

```
<svg version="1.1">
<polyline points="20,20 40,25 60,40 80,120 120,140 200,180"
  style="fill:none;stroke:black;stroke-width:3" />
</svg>
```

Path (větší objekt na jednom řádku)

Se zajímavým řešením přichází <path>, které se snaží vykreslovat složitější objekty tak, že všechno zapíšete na jeden řádek.
Aby jste určili o jaký konkrétní objekt se jedná, tak napíšete jedno z písmenek a pak klasicky souřadnice.

- M = křivka
- L = čára
- H = horizontální čára
- V = vertikální čára
- C = křivka
- S = hladká křivka
- Q = kvadratická křivka
- T = hladká kvadratická křivka
- A = Elipsa
- Z = ukončení




```
<svg version="1.1">
  <path id="lineAB" d="M 100 350 l 150 -300" stroke="red"
  stroke-width="3" fill="none">
  <path id="lineBC" d="M 250 50 l 150 300" stroke="red"
  stroke-width="3" fill="none">
  <path id="lineBC" d="M 175 200 l 150 0" stroke="red"
  stroke-width="3" fill="none">
  <path id="quadcurveABC" d="M 100 350 q 150 -300 300 0"
  stroke="blue" stroke-width="5" fill="none">
  <g stroke="black" stroke-width="3" fill="black">
    <circle id="pointA" cx="100" cy="350" r="3">
    <circle id="pointB" cx="250" cy="50" r="3">
```

```
    <circle id="pointC" cx="400" cy="350" r="3">
  </g>
  <g font-size="30" font="sans-serif" fill="black" stroke="none“ text-anchor="middle">
    <text x="100" y="350" dx="-30">A</text>
    <text x="250" y="50" dy="-10">B</text>
    <text x="400" y="350" dx="30">C</text>
  </g>
</svg>
```

Prosté texty? Ale zdaleka ne.

Vy si vážně myslíte, že SVG je jen obrázek? Ono to je i něco víc. No řekněte, který obrázek vám umožňuje vytvořit klikatelné texty jako odkazy?

Mám rád SVG!

```
<svg version="1.1">
  <text x="0" y="15" fill="red">Mám rád SVG!</text>
</svg>
```

I love SVG

```
<svg version="1.1">
  <text x="0" y="15" fill="red" transform="rotate(30
20,40)">I love SVG</text>
 </svg>
```

I love SVG I love SVG

```
<svg version="1.1">
   <defs>
    <path id="path1" d="M75,20 a1,1 0 0,0 100,0" />
  </defs>
  <text x="10" y="100" style="fill:red;">
    <textPath xlink:href="#path1">I love SVG I love
SVG</textPath>
  </text>
 </svg>
```

Klikni si!

```
<svg version="1.1">
  <a xlink:href="http://baraja.cz" target="_blank">
    <text x="0" y="15" fill="blue">Klikni si!</text>
  </a>
 </svg>
```

Čtete tuto knihu jako Ebook? V tom případě si klidně klikněte a bude to fungovat. Chtěl jsem to udělat i pro papírovou verzi, akorát mě nenapadlo rozumné řešení, jak toho dosáhnout.

To je o SVG všechno?

Zdaleka ne. Toto je jen pár ukázek těch nejpoužívanějších funkcí co se mi zrovna líbily. Některé ukázky SVG obrázků jsem převzal ze stránek http://w3schools.com, protože dělání vlastních nemá význam. Kdybych se měl pokusit udělat vlastní, tak by vlastně vypadali hodně podobně a nebo stejně. Pokud chcete vidět pár desítek ukázek na SVG, tak se koukněte sem: http://www.w3schools.com/svg/svg_examples.asp

ŽEBY KONEC?

HTML5 obsahuje spoustu nových funkcí a vlastností. Do budoucna jistě přenechává dobrý odkaz a smět vývoje je už víceméně známý. Mohl jsem tu napsat nějaký dojemný příběh, kde bych popisoval svoje zkušenosti, popisoval příběh o nějakém mladém chlapci, jak napsal svojí první „inteligentní“ aplikaci a stal se miliardářem.
V době psaní této knihy to jsou všechny možnosti, co současné HTML5 vývojářům nabízí. Podpora prohlížečů se už rapidně zlepšila, ale ještě to je běh na dlouho trať a vyplatí se ještě rok nebo dva počkat. Do té doby se snad vše naučíte a já vydám další vydání :D
Jestli si čtete tento závěrečný článek, tak už je vidět, že vás můj styl psaní baví a chcete vědět víc. A za to jsem rád!
Knihu jsem tvořil postupně, nemám z toho žádné peníze, dělám to pro radost, nikoliv pro výdělek. Pokud vás tedy kniha zaujala, tak ocením když na vaší webovou stránku vložíte odkaz na můj web (http://baraja.cz) a nebo o tomto projektu napíšete hezký článek do vašeho blogu.
Doufám že oceníte mojí práci, strávil jsem na tom desítky hodin času a musel jsem všechno vymyslet. Mě to nikdo nenaučil, neměl jsem k dispozici jednoduché články. Všechno jsem musel pochopit z anglického, stále nehotového manuálu a většinu věcí jsem musel odvodit a pokusit se na to přijít metodou „pokus-omyl“.

Ještě poznámka na závěr:

Knížka je sice moje a mám na ní všechny autorské práva, ale já jsem se jich částečně vzdal. Když už knížku upravíte, budete dále šířit a nebo od základu přepíšete, tak nezapomeňte uvést mé jméno jako autora. Vám to nic neudělá a mě tím vyjádříte vaší podporu v této oblasti.

Další vydání

Toto je první vydání, které vyšlo **1.5.2012**. Pokud to čtete po delší době, tak některé informace nemusejí být aktuální a raději navštivte oficiální dokumentaci na adrese http://www.w3.org/TR/html5-diff/ (anglicky)

Do budoucna bych chtěl vydat pokračování a nebo knihu o jiném jazyku. Třeba o PHP6 (které bude za několik let), o CSS3 a podobně. V případě velkého zájmu se pustím do psaní.

Děkuji,
Jan Barášek (Autor) – Též přezdíván: Baraja